Amendments to the Claims

Please amend Claims 1, 9, 17, 25 and 26. The Claim Listing below will replace all prior versions of the claims in the application:

Claim Listing

1. (Currently amended) A collector for collecting non-referenced objects stored in a heap by a program executing in a computer system comprising:

an object allocation routine which stores an object of a particular type in one of a plurality of logical partitions in the heap dependent on a predefined category ~~for~~ assigned to the object type, such that each object of a certain category is stored in one logical partition of the heap and objects of a category different from the certain category are stored in a logical partition different from the one logical partition; and

a collection routine which searches one of the logical partitions of the heap for ~~referenced~~ objects to which references are made and reclaims non-referenced objects stored in the searched logical partition of the heap.

2. (Original) The collector as claimed in Claim 1 further comprising:

a sample and partition routine which defines a category of an object stored in the heap to be hot or cold.

3. (Previously presented) The collector as claimed in Claim 2 wherein upon determining that a hot logical partition is full, the collection routine searches a cold logical partition and the hot logical partition for referenced objects and moves referenced objects of the hot category stored in the hot logical partition to the cold logical partition.

4. (Previously presented) The collector as claimed in Claim 2 wherein the sample and partition further comprises:

a write barrier elimination routine, which eliminates a write barrier for an intergenerational pointer between an object stored in a hot logical partition and an object stored in a cold logical partition.

5. (Original) The collector as claimed in Claim 4 wherein the write barrier elimination routine eliminates a write barrier by replacing a write barrier machine code instruction with a no operation machine code instruction.

6. (Original) The collector as claimed in Claim 2 wherein the sample and partition routine defines the object category dependent on object type mortality.

7. (Original) The collector as claimed in Claim 6 wherein the sample and partition routine estimates the object mortality dependent on difference of the number of bytes of the object type stored in the heap before a collection and the number of bytes of the object type stored in the heap after the collection.

C1 8. (Previously presented) The collector as claimed in Claim 2 wherein the sample and partition routine partitions the heap to minimize intergenerational pointers between a hot logical partition and a cold logical partition.

9. (Currently amended) A collector for collecting non-referenced objects stored in a heap by a program executing in a computer system comprising:

means for storing an object of a particular type in one of a plurality of logical partitions in the heap dependent on a predefined category ~~for~~ assigned to the object type, such that each object of a certain category is stored in one logical partition of the heap and objects of a category different from the certain category are stored in a logical partition different from the one logical partition;

means for searching one of the logical partitions in the heap for referenced objects; and

means for reclaiming non-referenced objects stored in the searched logical partition.

10. (Previously presented) The collector as claimed in Claim 9 further comprising:

means for partitioning the heap into a cold logical partition and a hot logical partition by defining a category of an object stored in the heap to be hot or cold.

11. (Previously presented) The collector as claimed in Claim 10 wherein upon determining that a hot logical partition is full, the means for searching searches a cold logical partition and the hot logical partition for referenced objects and moves referenced objects stored in the hot logical partition to the cold logical partition.

12. (Previously presented) The collector as claimed in Claim 10 wherein the means for partitioning further comprises:

means for eliminating a write barrier for an intergenerational pointer between an object stored in the hot logical partition and an object stored in the cold logical partition.

C1

13. (Original) The collector as claimed in Claim 12 wherein the means for eliminating a write barrier replaces write barrier machine code instructions with no operation machine code instructions.

14. (Original) The collector as claimed in claim 10 wherein the means for partitioning defines a hot object dependent on object type mortality.

15. (Original) The collector as claimed in Claim 14 wherein the means for partitioning estimates the object mortality dependent on difference of the number of bytes of the object type stored in the heap before a collection and the number of bytes of the object type stored in the heap after the collection.

16. (Previously presented) The collector as claimed in Claim 10 wherein the means for partitioning partitions the heap to minimize intergenerational pointers between the hot logical partition and the cold logical partition.

17. (Currently amended) A method for collecting non-referenced objects stored in a heap by a program executing in a computer system comprising the steps of:

storing an object of a particular type in one of a plurality of logical partitions in the heap dependent on a predefined category ~~for~~ assigned to the object type, such that each object of a certain category is stored in one logical partition of the heap and objects of a category different from the certain category are stored in a logical partition different from the one logical partition;

searching one of the logical partitions of the heap for referenced objects; and

reclaiming non-referenced objects stored in the searched logical partition.

C1

18. (Previously presented) The method as claimed in Claim 17 further comprising the step of: partitioning the heap into a cold logical partition and a hot logical partition by defining hot logical partition objects and cold logical partition objects.

19. (Previously presented) The method as claimed in Claim 18 wherein upon determining that the hot logical partition is full, the step of reclaiming further comprises the step of:

moving referenced objects stored in the hot logical partition to the cold logical partition.

20. (Previously presented) The method as claimed in Claim 18 wherein the step of partitioning further comprises the step of:

eliminating a write barrier for an intergenerational pointer between an object stored in the hot logical partition and an object stored in the cold logical partition.

21. (Original) The method as claimed in Claim 20 wherein the step of eliminating a write barrier replaces write barrier machine code instructions with no operation machine code instructions.

22. (Original) The method as claimed in claim 18 wherein the step of partitioning further comprises the step of:

identifying a hot object dependent on object type mortality.

23. (Original) The method as claimed in Claim 22 wherein the step of identifying estimates the object type mortality dependent on difference of the number of bytes of the object type stored in the heap before a collection and the number of bytes of the object type stored in the heap after the collection.

24. (Previously presented) The method as claimed in Claim 18 wherein the step of partitioning partitions the heap to minimize intergenerational pointers between the hot logical partition and the cold logical partition.

25. (Currently amended) A computer system comprising:

a central processing unit connected to a memory bus by a system bus;

an I/O system, connected to the system bus by a bus interface; and

a collector for collecting non-referenced objects stored in a heap by a program executing in a computer system, the collector:

storing an object of a particular type in one of a plurality of logical partitions in the heap dependent on a predefined category ~~for~~ assigned to the type, such that each object of a certain category is stored in one logical partition of the heap and objects of a category different from the certain category are stored in a logical partition different from the one logical partition;

searching one of the logical partitions for referenced objects; and

reclaiming non-referenced objects stored in the searched logical partition.

26. (Currently amended) A computer program product for collecting non-referenced objects stored in a heap by a program executing in a computer system, the computer program product comprising a computer usable medium having computer readable program code thereon, including program code which:

stores an object of a particular type in one of a plurality of logical partitions in the heap dependent on a predefined category ~~for~~ assigned to the type, such that each object of a certain category is stored in one logical partition of the heap and objects of a category different from the certain category are stored in a logical partition different from the one logical partition;

searches one of the logical partitions for referenced objects; and

reclaims non-referenced objects stored in the searched logical partition.